UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DEFINICIÓN E IMPLEMENTACIÓN DE UNA API REST PARA SISTEMAS DE RECOMENDACIÓN

Iván García Rodríguez
Tutor: Alejandro Bellogín Kouki
Ponente (si procede): Iván Cantador Gutiérrez

Julio 2018

DEFINICIÓN E IMPLEMENTACIÓN DE UNA API REST PARA SISTEMAS DE RECOMENDACIÓN

AUTOR: Iván García Rodríguez
TUTOR: Alejandro Bellogín Kouki

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2018

Resumen (castellano)

En la actualidad, los Sistemas de Recomendación son un activo importante en el mundo interactivo de internet con el usuario. El fin del Sistema de Recomendación con respecto al usuario es ofrecer una experiencia de uso cómoda e intuitiva, siendo capaz de ofrecerle recomendaciones que le puedan ser útiles y no estar perdido por la web. Sin embargo, el servicio que proporciona el sistema tiene como objetivo recaudar clientes y así obtener el máximo beneficio posible. Este servicio se ha incrementado potencialmente en los últimos años, por tanto, los Sistemas de Recomendación seguirán creciendo y acaparando más importancia en el mercado tecnológico.

Este Trabajo Fin de Grado se basa principalmente en la implementación de una API REST basada en un Sistema Recomendación. Para ello se hará un estudio previo para conocer un poco mejor el mundo de los Sistemas de Recomendación, el objetivo que tienen, las distintas técnicas y evaluaciones de las recomendaciones existentes, sus aplicaciones más populares y el impacto que tienen en la sociedad más en profundidad.

A continuación, se realizará el diseño y desarrollo de esta API REST con la que se quiere demostrar que se puede crear un servicio web adaptado a un Sistema de Recomendación teniendo una estructura simple basada en usuarios, ítems y eventos. Se podrán generar recomendaciones para el usuario gracias a la utilización de librerías externas y en las que se utilizarán algoritmos de filtrado colaborativo.

Por último, se creará una pequeña interfaz de usuario para poder probar la API REST de manera gráfica para poder testearla, así como las pruebas unitarias para poder garantizar el correcto funcionamiento del servidor.

Palabras clave (castellano)

Sistemas de Recomendación, usuarios, ítems, eventos, servidor, cliente, interfaz, API REST, filtrado colaborativo.

Abstract (English)

Nowadays, Recommendation Systems are an important asset in the interactive internet world with the user. The goal of the Recommendation System with respect to the user is to offer a comfortable and intuitive experience, being able to offer recommendations that can be useful and avoid being lost in the web. However, the service provided by these systems is intended to raise customers and thus obtain the maximum profit possible. This service has increased potentially in recent years; therefore, the recommendation systems will continue to grow and gain more importance in the technological market.

This Bachelor Thesis is mainly based on the implementation of a REST API based on a recommendation system. This will be done, first, by going through a preliminary study to know better about the world of the recommendation systems, their goals, the different available techniques and how they are evaluated, their most popular applications, and the impact they have on the society.

Next, the design and development of this REST API is presented, where we want to demonstrate how a Web service adapted to a recommendation system having a simple structure based on users, items and events is created. Such a service will be able to generate recommendations for the user thanks to the use of external libraries, where collaborative filtering algorithms will be exploited.

Finally, a user interface will be created to be able to test the REST API in a graphical way, which, together with the developed unit tests, guarantee the correct functioning of the implemented framework.

Keywords (inglés)

Recommendation Systems, users, items, events, server, client, interface, REST API, collaborative filtering.

Agradecimientos

En primer lugar, agradecer a mi familia todos estos años que han estado a mi lado, y en los que siempre me han dado su cariño. En especial, agradecer a mi madre, todo el amor que me regala cada día y la confianza que me transmite en todo momento, de lo cual estaré eternamente agradecido.

Agradecer a mis amigos todo el aguante que tienen conmigo y darle las gracias por la amistad que nos une. Suren, Dani, Guille, Kamil, Iván, gracias por tantos años juntos y espero estar mucho tiempo más a vuestro lado.

Fran, Kiko, mis dos descubrimientos de la universidad, y dos grandes amigos que se que me llevo de estos años en la carrera, muchísimas gracias.

Aunque lleve poco tiempo, querría dar las gracias a la empresa que me acogió en las prácticas, y que ahora han decidido contar conmigo. Un grupo de personas fantástico en el que me han integrado fenomenalmente desde el primer día. Alberto, Ernesto, muchas gracias por confiar en mí.

Y, por último, me gustaría dar las gracias a mi tutor de este trabajo, Alejandro, del cual me siento muy agradecido por la calma que me ha transmitido en otro momento y toda la ayuda que me ha ofrecido.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

En el mundo de hoy es innumerable la cantidad de productos que nos podemos encontrar en la web, a la vez que crece el interés de los usuarios de la web, por eso es necesario que Internet ofrezca lo que puede llamar la atención al usuario porque si no, puede no sentirse cómodo en la navegación por internet al no ser capaz de encontrar nada que le atraiga y satisfaga sus necesidades a la vez. Pero esto no influye solo al usuario que navega en la web y desea, por ejemplo, comprar un libro, también influye de manera notoria en el proveedor del servicio web, ya que, si este es capaz de saber qué productos son de interés para el cliente, se puede asegurar que el usuario vuelva a entrar en la página web a seguir buscando productos o llegar incluso a adquirirlo, consiguiendo así beneficio para la empresa.

Para esto entran en juego los Sistemas de Recomendación, los cuales deben ser capaces de leer los movimientos o transacciones por la web del usuario, saber cuáles son las páginas más frecuentadas por éste o cuáles son sus preferencias o valoraciones sobre ítems. Habitualmente los Sistemas de Recomendación suelen ubicarse en el comercio online, y donde podemos pensar que más interés puedan llegar a tener las recomendaciones para conseguir mayor número de ventas, como podría ser Amazon. Pero, sin embargo, en ámbitos como servicios de vídeo en streaming (Netflix), o servicios online de música (Spotify), también deben ser capaces de garantizar al usuario una gran experiencia de uso y una comodidad a la hora de ofrecer al usuario películas o canciones que sean de su agrado, según, por ejemplo, sus últimas reproducciones o películas o canciones más valoradas, de manera que permita una experiencia más agradable al usuario y un uso continuado del sistema o incluso una futura suscripción.

Los Sistemas de Recomendación, en definitiva, son una alternativa a solicitar opiniones a conocidos o a expertos sobre un ámbito cuando se va a tomar una decisión para adquirir algún ítem a través de la web, y no se tiene mucha información sobre el producto o sobre el ámbito en el que se esté moviendo el usuario.

A la par, el número de servicios web ha incrementado de manera notable, así como su desarrollo, y para ello se está utilizando cada vez más la arquitectura REST. Esta arquitectura es más simple y ligera que otras que por ejemplo las arquitecturas SOA o SOAP. Por ello, en este TFG nos planteamos la posibilidad de utilizar esta arquitectura en el contexto de los Sistemas de Recomendación.

1.2 Objetivos

En el anterior punto se ha hablado de los Sistemas de Recomendación, así como de los servicios web basados en API REST, y en este trabajo de final de grado el objetivo es diseñar un acercamiento a un Sistema de Recomendación basado en API REST de manera sencilla y que pueda cumplir con las necesidades básicas tanto para el usuario como para el desarrollador de un Sistema de Recomendación.

Además, también se analizará más en detalle el comportamiento de un Sistema de Recomendación, así como las distintas técnicas de recomendación dentro del sistema.

Todo esto será posible con el desarrollo de un servicio web, por tanto, también veremos que con una arquitectura simple como la de una API REST, se puede implementar un sistema que ofrezca las mejores opciones a distintos usuarios con distintas preferencias.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1: Motivación y objetivos del proyecto. Este capítulo es en el que estamos actualmente. En este se ha descrito cual es la motivación de este trabajo y cuáles son los objetivos finales.
- Capítulo 2: Estado del arte. El capítulo siguiente se basa en explicar las distintas técnicas de recomendación, las distintas maneras de implementar un servicio web, y, sobre todo, herramientas y metodologías para implementar un Sistema de Recomendación a través de una API REST.
- Capítulo 3: Análisis. En el capítulo 3 se analizan los requisitos funcionales y no funcionales que tiene el sistema que se va a implementar, según los objetivos descritos.
- Capítulo 4: Diseño. Este capítulo trata sobre el diseño se va a utilizar según la estructura de una API REST y su manera de realizar cada llamada al servicio web.
- Capítulo 5: Desarrollo. En el capítulo 5 se detalla el método de desarrollo utilizado como el lenguaje de programación utilizado, el entorno de programación utilizado para hacer más fácil la construcción del código, las librerías utilizadas, y el software en el que apoyarse para montar un servicio web.
- Capítulo 6: Pruebas. En el sexto capítulo, se muestran pruebas unitarias para poder comprobar el resultado correcto del Sistema de Recomendación y de la correcta integración del servicio web.
- Capítulo 7: Conclusión. En este último capítulo, se resumen las conclusiones de montar un Sistema de Recomendación basado en una API REST y de lo que ha supuesto realizar este trabajo de final de grado.

2 Estado del arte

2.1 Impacto en la sociedad

Como se ha dicho antes, el estudio de los Sistemas de Recomendación (SRs) es relativamente nuevo comparado con la investigación en otras herramientas clásicas de sistemas de información (p.e., bases de datos o motores de búsqueda). Los Sistemas de Recomendación surgen como una investigación independiente alrededor de los años 90. En los últimos años, el interés por los SRs ha incrementado dramáticamente, como indican algunos de los siguientes sucesos:

1. Los Sistemas de Recomendación juegan un papel muy importante en páginas web de Internet sumamente valoradas como Amazon, YouTube, Netflix, Spotify, LinkedIn, Facebook, Tripadvisor, y Last.fm. Además, muchas empresas de medios de comunicación ahora desarrollan y despliegan SRs como parte de los servicios que ellos proporcionan a sus seguidores. Un dato curioso es que, por ejemplo, Netflix, el proveedor de "streaming media" que tiene más demanda actualmente, concedía en el año 2006 un premio de un millón de dólares a aquel primer equipo que consiguiera mejorar un 10% el rendimiento de su Sistema de Recomendación.
2. Hay conferencias y talleres dedicados expresamente al ámbito, como la "Association of Computing Machinery's (ACM)" sobre Sistemas de Recomendación, establecida en 2007. Esta conferencia se sitúa como el primer acontecimiento anual en la investigación de la tecnología de recomendaciones y sus usos. Además, las sesiones dedicadas a SRs son con frecuencia incluidas en conferencias más tradicionales en el área de bases de datos, sistemas de información y sistemas adaptables.
3. En las instituciones de enseñanza superior en el mundo entero, el estudiante y los cursos graduados son dedicados completamente a Sistemas de Recomendación, los tutoriales sobre SRs son muy populares en conferencias de informática y un libro que trata las técnicas de los SRs ha sido publicado también.
4. Hubo varias publicaciones especiales en diarios académicos que cubren la investigación y el desarrollo en el ámbito de los Sistemas de Recomendación [11]. Entre las revistas que han dedicado publicaciones a los SRs están las siguientes: AI Communications; IEEE Intelligent Systems; International Journal of Electronic Commerce; ACM Transactions on Computer Human Interaction; User Modeling and User-Adapted Interaction; y ACM Transactions on Intelligent Systems and Technology.

2.2 Función de los Sistemas de Recomendación

En la sección anterior, definimos SRs como instrumentos de software y técnicas que proveen a los usuarios de sugerencias para artículos que pueden desear utilizar.

Ahora deseamos refinar esta definición para mostrar la gama de papeles que puede jugar un SR. En primer lugar, debemos distinguir entre el papel jugado por el Sistema de Recomendación por parte del proveedor del servicio, desde él hasta el usuario del sistema.

Por ejemplo, un Sistema de Recomendación de viajes es normalmente presentado por un intermediario de viajes como puede ser Expedia.com, o una organización de eventos, como Visitfinland.com, para aumentar su volumen de ventas o vender más habitaciones de hotel (en el caso del intermediario), y aumentar el número de turistas al destino (en el caso de la organización de eventos). Las motivaciones primarias del usuario para tener acceso a los dos sistemas ilustrados deberían ser encontrar un hotel conveniente y acontecimientos interesantes, o algún tipo de atracción visitando un destino.

De hecho, hay varias razones en cuanto a por qué los proveedores de servicio pueden querer explotar esta tecnología [11][2]:

- Aumento del número de artículos vendidos. Esta es probablemente la función más importante para un Sistema de Recomendación comercial, p.e., ser capaz de vender un conjunto de ítems adicional comparando con lo que es vendido sin cualquier tipo de recomendación. Este objetivo es alcanzado porque los artículos recomendados seguramente satisfacen las necesidades del usuario.
  Presumiblemente el usuario reconocerá estos después de haber intentado varias recomendaciones. Aplicaciones que no tienen un fin comercial tienen fines similares, incluso si no hay ningún coste para el usuario que es asociado a la selección de un ítem. Por ejemplo, el contenido de una red apunta al aumento del número de noticias leídas sobre su sitio. En general, podemos decir que, desde el punto de vista del proveedor de servicios, el objetivo primario para introducir un SR es de aumentar la tasa de conversión.
- Venta de más variedad de artículos. Otra función principal de un SR es permitir al usuario seleccionar los ítems que podrían ser difíciles de encontrar sin una recomendación exacta.
- Aumento de la satisfacción del usuario. Un Sistema de Recomendación bien diseñado también es capaz de mejorar la experiencia del usuario con la página web o la aplicación. El usuario encontrará recomendaciones interesantes, relevantes y, con una interacción diseñada adecuadamente, también disfrutará usando el sistema. La combinación de recomendaciones eficaces, exactas y una interfaz usable aumentarán la evaluación personal del usuario sobre el sistema. Esto, a su vez, aumentará el uso del sistema y la probabilidad de que las recomendaciones sean aceptadas.
- Aumento de la fidelidad del usuario. Un usuario compensaría ser leal a una página web, si cuando la visita, ésta reconoce al antiguo cliente y lo trata como un visitante valorado. Esto es una característica normal de un SR, ya que muchos SRs calculan recomendaciones, lo que permite potenciar la información adquirida del usuario durante interacciones previas, como las valoraciones de los artículos. En consecuencia, cuanto más duradera sea la interacción del usuario con la página web, más refinado será el modelo del usuario.
- Mejor comprensión de lo que el usuario desea. Otra función importante de un Sistema de Recomendación, la cual puede ser aprovechada en otras muchas aplicaciones, es la descripción de las preferencias del usuario, que son recogidas explícitamente o predichas por el sistema. El proveedor de servicios entonces puede decidir reutilizar este conocimiento para otros objetivos, como puede ser la mejora del control de ventas o producción del artículo. Por ejemplo, en un SR turístico, las

organizaciones de viajes pueden anunciar la organización de un evento específico a nuevos sectores de un cliente, o anunciar un mensaje promocional sacado por el análisis de los datos recolectados por el SR (transacciones de los usuarios).

2.3 Tareas populares de un Sistema de Recomendación

Un Sistema de Recomendación debe equilibrar las necesidades de un proveedor de servicio, y las de un usuario que también pueda necesitar un RS, si éste le ayudara a conseguir sus tareas u objetivos, y ofrecer un servicio que sea valiosos para ambos.

J.L. Herlocker en [8], en un artículo que se ha convertido en una referencia clásica en este ámbito de los SRs, define once tareas populares a las cuales un RS puede contribuir a su realización. Unas pueden ser consideradas como las tareas principales que normalmente son asociadas con un SR, y las otras podrían ser consideradas como modos más "oportunistas" de explotar un SR. Su función primaria es localizar los ítems más relevantes para la necesidad del usuario, pero también puede usarse, por ejemplo, para comprobar la importancia de una página web, o para descubrir varios usos de una palabra en un conjunto de documentos.

- Encontrar algunos ítems buenos. Recomendar al usuario algunos artículos buenos como una lista valorada con las predicciones de cuánto le gustarían al usuario. Esta es la tarea de recomendación principal de muchos sistemas comerciales. Algunos sistemas no muestran la predicción predicha.
- Encontrar todos los ítems buenos. Recomendar todos los artículos que puedan satisfacer las necesidades del usuario. En estos casos es insuficiente con encontrar algunos artículos buenos. Esto es así cuando el número de ítems es pequeño. En estas situaciones, además del beneficio derivado por una cuidadosa investigación de todas las posibilidades, el usuario también puede beneficiarse del "ranking" del SR de estos artículos, o de explicaciones adicionales.
- Anotaciones en contexto. Considerando un contexto existente, el Sistema de Recomendación resalta algunos ítems dependiendo de las preferencias del usuario a lo largo del tiempo.
- Recomendar una secuencia. En vez de enfocar la generación de recomendaciones como una única recomendación, la idea es que se genere una secuencia de recomendaciones de ítems que complace en total al usuario.
- Recomendar un bloque. Sugerir un grupo de ítems que son similares entre ellos. Por ejemplo, en un plan de viajes, este bloque podría estar compuesto por lugares de atracción, destinos, hoteles que están localizados en un área delimitada.
- *Solamente hojeando*. En esta tarea, el usuario navega por el catálogo sin la intención inminente de obtener un artículo. La tarea del Sistema de Recomendación es de ayudar al usuario a hojear aquellos ítems que con mayor probabilidad sean del interés del usuario para aquella sesión. Esta es una tarea que también ha sido apoyada por técnicas de hipermedios de comunicación adaptables.
- Hacer un SR confiable. Algunos usuarios no confían en los Sistemas de Recomendación, así que juegan con ellos para ver cómo de buenos son en la construcción de recomendaciones. De ahí, un cierto sistema también puede ofrecer

funciones específicas para dejar a los usuarios probar su comportamiento, además de aquellos solamente requeridos para obtener recomendaciones.

- Mejorar el perfil. Esto se relaciona con la capacidad del usuario de proporcionar la información al SR sobre sus gustos o sobre lo que no le agrada. Esta es una tarea fundamental que es estrictamente necesaria para proporcionar recomendaciones personalizadas. Si el sistema no tiene de primeras ningún conocimiento sobre los gustos del usuario activo, entonces sólo podrá proporcionar las mismas recomendaciones que le daría a un usuario normal.
- Expresarse. Algunos usuarios no se preocupan por las recomendaciones en absoluto. Más bien lo que es importante para dichos usuarios es que ellos puedan contribuir con sus valoraciones y expresen sus opiniones. La satisfacción del usuario para esta tarea puede causar la lealtad continuada del usuario al uso de la web o aplicación.
- Ayudar a los demás. Algunos usuarios se contentan con poder ayudar aportando información, como, por ejemplo, valorando los artículos, porque creen que la comunidad se beneficia de su contribución.
- Influenciar a los demás. En webs basadas en SRs, hay usuarios que tienen como objetivo principal influir en otros usuarios en la compra de productos particulares. En realidad, también hay usuarios no tan buenos, que puedan usar el sistema para penalizar ciertos ítems.

Como indican estas tareas, el papel de un Sistema de Recomendación dentro de un sistema de información puede ser bastante diverso. Esta diversidad obliga a la utilización de una serie de técnicas y fuentes de conocimiento diferentes.

2.4 Datos y fuentes de conocimiento

Los SRs son sistemas informáticos que rápidamente combinan varias clases de datos para construir sus recomendaciones. Los datos son principalmente sobre los artículos que se van a sugerir y los usuarios que recibirán dichas recomendaciones. Pero, ya que los datos y fuentes de conocimiento disponibles para estos sistemas pueden ser muy diversos, en última instancia pueden ser explotados según la técnica de recomendación.

En general, hay técnicas de recomendación que son pobres en conocimiento, concretamente, las que usan datos muy simples y básicos, como valoraciones de usuario o evaluaciones de ítems. Otras técnicas son muchos más dependientes del conocimiento. En ellas se usan descripciones ontológicas de los usuarios o de los artículos, limitaciones, o relaciones sociales y actividades de los usuarios. En cualquier caso, como una clasificación general, los datos usados por SRs se refieren a tres tipos de objetos: ítems, usuarios y transacciones, es decir, la relación entre usuario e ítem.

Las valoraciones son la forma más popular de transacción de datos que un SR recoge. Estas valoraciones pueden ser recogidas explícitamente o implícitamente. En la colección explicita de valoraciones, se pide al usuario proporcionar una opinión sobre un artículo por una medida de evaluación. Las valoraciones pueden ser de varias formas:

- Valoraciones numéricas. Por ejemplo, evaluar un artículo del 1 al 5.
- Valoraciones ordinales. Como, “muy de acuerdo”, “de acuerdo”, “neutral”, “en desacuerdo”, “muy en desacuerdo”.
- Valoraciones binarias. Solo piden al usuario si un artículo está bien o mal.
- Valoraciones singulares. Indican que un usuario ha observado o comprado un artículo.

Otra manera de evaluación de los usuarios consiste en etiquetas asociadas por los usuarios con los ítems que el sistema presenta.

En las transacciones que recogen valoraciones implícitas, los objetivos del sistema son deducir la opinión del usuario basándose en las acciones del usuario. Por ejemplo, si un usuario busca en Amazon por “Yoga”, se proporcionará una larga lista de libros. A cambio, el usuario puede pulsar sobre un cierto libro para recibir información adicional. En este punto, el sistema puede deducir que el usuario está algo interesado en ese libro.

2.5 Técnicas de Recomendación

Content-Based. El sistema aprende a recomendar los ítems que son similares a algunos que al usuario ya le gustaron en el pasado. La semejanza de artículos es calculada basándose en los rasgos asociados a los artículos comprados. Por ejemplo, si un usuario ha valorado positivamente una película que pertenece al género de la comedia, entonces el sistema aprende a recomendar otras películas de este género [4].

Las técnicas de recomendación clásicas basadas en el contenido apuntan a la correspondencia de los atributos del perfil del usuario contra los atributos de los artículos. En la mayoría de los casos, los atributos de los artículos son simplemente las palabras clave que son extraídas de las descripciones de los artículos. Técnicas de indexación semántica representan el artículo y los perfiles de usuario usando conceptos en vez de palabras clave.

Se presentan dos grupos principales de técnicas de indexación semántica: descendente y ascendente. Las técnicas descendentes confían en la integración de fuentes de conocimiento externas, mientras que las técnicas ascendentes confían en una representación semántica de peso ligero basada en la hipótesis de que el significado de palabras depende de su uso en la gran parte del cuerpo de documentos textuales.

Collaborative Filtering. La puesta en práctica original y más simple de este enfoque hace recomendaciones al usuario activo basándose en ítems que a otros usuarios con gustos similares les gustó en algún momento. La similitud en el gusto de dos usuarios es calculada guiándose en la semejanza de las valoraciones históricas de los usuarios. Esta es la razón por la cual se refiera a la filtración colaborativa como la “correlación de gente a gente”. El filtrado colaborativo es considerado como la técnica de implementación más popular y general en los Sistemas de Recomendación, ya que no necesita que los ítems tengan atributos u otros procesamientos externos [6][9].

Demographic. Este tipo de sistema recomienda artículos basados en el perfil demográfico del usuario. La suposición es que recomendaciones diferentes deben ser generadas para diferentes lugares demográficos. Muchas páginas web adoptan soluciones simples y eficaces de personalización basadas en datos demográficos. Por ejemplo, los usuarios son

enviados a páginas web determinadas según su lengua o país. O las sugerencias pueden ser adaptadas según la edad del usuario. Mientras estos criterios han sido bastante populares en la literatura de márketing, hubo relativamente poca investigación de SR apropiada sobre sistemas demográficos [2].

Knowledge-Based. Los sistemas basados en conocimiento recomiendan artículos basándose en el conocimiento de un dominio específico sobre cómo ciertas características de un ítem se relacionan con las necesidades de los usuarios y preferencias, en última instancia, sobre cómo de útil es el ítem para el usuario. Importantes SRs basados en conocimientos son los casos reales. En estos sistemas, una función similar estima las necesidades del usuario (descripción del problema) y emparejan las recomendaciones (las soluciones del problema). Aquí, la "puntuación" de similitud puede ser interpretada como la utilidad de la recomendación para el usuario. Sistemas basados en conocimiento tienden a trabajar mejor que otros al principio de su despliegue, pero si no están equipados con el estudio de los componentes, éstos pueden ser sobrepasados por otros métodos que pueden explotar la interacción entre el humano y el ordenador [3].

Los sistemas basados en limitaciones son otro tipo de SR basado en el conocimiento. En términos del conocimiento, ambos sistemas son similares: las exigencias del usuario son recogidas, automáticamente se proponen correcciones para dichas exigencias incoherentes en situaciones donde puede que no se encuentren soluciones, y los resultados de la recomendación son explicados. La principal diferencia es cómo se calcula el camino hacia las soluciones. En los casos reales de recomendación se determinan las recomendaciones sobre la métrica basada en la similitud, mientras que en los sistemas basados en limitaciones predominantemente se explotan las bases de conocimiento predefinidas que contienen reglas explicitas sobre cómo relacionar exigencias de cliente con rasgos del ítem.

Community-Based. Este tipo de sistema recomienda artículos a un usuario basándose en las preferencias de sus amigos. "Dime con quién andas, y te diré quién eres". Muchas pruebas apuntan a que la gente tiende a confiar más en las recomendaciones de sus amigos que en recomendaciones de individuos similares, pero desconocidos. Esta observación, combinada con la popularidad creciente de las redes sociales, genera un mayor interés por sistemas basados en una comunidad de usuarios. Este tipo de Sistema de Recomendación adquiere y modela la información sobre las relaciones sociales de los usuarios y las preferencias de los amigos del usuario. La recomendación está basada en las valoraciones que han hecho los amigos del usuario sobre los ítems. De hecho, estos SRs continúan con el auge de las redes sociales y permiten una adquisición simple y comprensiva de datos relacionados con las relaciones sociales de los usuarios.

Hybrid Recommender Systems. Estos Sistemas de Recomendación están basados en la combinación de las técnicas anteriormente mencionadas. Un sistema mixto que combina las técnicas A y B, intenta usar las ventajas de A para compensar las desventajas de B. Por ejemplo, los métodos de CF (Collaborative Filtering) tienen problemas con nuevos artículos, ya que no pueden recomendar artículos que no tienen valoraciones. Esto no limita el criterio de la técnica "content-based" ya que la predicción para nuevos artículos está basada en su descripción (rasgos/características) que están normalmente disponibles.

2.6 Aplicaciones de los Sistemas de Recomendación

La investigación de los Sistemas de Recomendación, aparte de su contribución teórica, está generalmente unido a mejorar la práctica industrial de los SRs y se implica en la investigación sobre varios aspectos prácticos que se aplican a la puesta en marcha de estos sistemas. De hecho, un SR es un ejemplo de uso de "machine learning" a gran escala, así como de algoritmos de minería de datos en la práctica comercial. El interés común en el ámbito, por parte de la investigación de la comunidad y de la industria, ha aprovechado la disponibilidad de datos para la investigación, por un lado, y el desarrollo de algoritmos mejorados por otra parte.

La investigación práctica en los SRs examina los aspectos que son relevantes a etapas diferentes en el ciclo de vida de un SR, concretamente, el diseño del sistema, su puesta en práctica, evaluación, mantenimiento y mejora durante la operación del sistema. El Premio de Netflix anunciado en 2006, era un importante evento para la comunidad y la industria de investigadores de los Sistemas de Recomendación, y su interacción mutua. Esto destacó la importancia de la recomendación de artículos a usuario y aceleró el desarrollo de nuevas técnicas de recomendación de minería de datos. Incluso aunque el Premio de Netflix iniciara muchas actividades de investigación, el premio era una simplificación del problema de recomendación completo. Esto consistió en predecir las posiciones del usuario optimizando la métrica "Root Mean Square Error (RMSE)" entre las valoraciones predichas y las reales. El primer factor para considerar cuando se diseña un SR es el ámbito de la aplicación ya que esto tiene un efecto principal sobre el criterio de elección del algoritmo. Montaner en [10] proporciona una taxonomía de SRs y clasifican existentes aplicaciones de Sistema de Recomendación por sus dominios específicos.

Basándose en estos dominios de aplicación específicos, las clases más generales para los usos de Sistemas de Recomendación son:

- Entretenimiento. Recomendación para películas, música, juegos e IPTV (Internet Protocol Television).
- Contenido. Periódicos personalizados, recomendación para documentos, recomendaciones de páginas web, usos de enseñanza virtual, y los filtros de correo electrónico.
- Comercio "online". Recomendaciones de productos para comprar como libros, cámaras, ordenadores, etc. para consumidores.
- Servicios. Las recomendaciones de agencias de viajes, recomendación de expertos de consultoría o recomendaciones de casas para alquilar.
- Social. Recomendación de gente en redes sociales, y las recomendaciones de contenido de medios de comunicación como tuits, comentarios de Facebook, actualizaciones de LinkedIn, etc.

2.7 Evaluación de los Sistemas de Recomendación

La búsqueda de Sistemas de Recomendación está siendo dirigida con un fuerte énfasis sobre la práctica y usos comerciales. Una cuestión muy importante relacionada con el lado práctico del desarrollo de un Sistema de Recomendación es la necesidad de evaluar la

calidad y el valor de estos sistemas. La evaluación es requerida en las distintas etapas del ciclo de vida del sistema para varios objetivos [7][8]. A la hora del diseño, la evaluación se requiere para verificar la selección del criterio de recomendación apropiado. En esta fase, la evaluación debe ser implementada "offline" y los algoritmos de recomendación son comparados con las interacciones de usuario almacenadas. Una evaluación "offline" consiste en controlar varios algoritmos sobre los mismos conjuntos de datos de las interacciones del usuario y comparar su funcionamiento. Este tipo de evaluación por lo general se aplica sobre datos públicos de referencia si los datos apropiados están disponibles. El diseño de los experimentos "offline" deberían seguir prácticas de diseño experimental conocidas para asegurar resultados confiables. Los experimentos offline pueden medir la calidad del algoritmo escogido en la realización de su tarea de recomendación. Sin embargo, tal evaluación no puede proporcionar ninguna clarividencia sobre la satisfacción del usuario, la aceptación o la experiencia con el sistema. Los algoritmos podrían ser muy exactos en la solución del problema de recomendación principal, pero por alguna otra razón el sistema puede no ser aceptado por usuario, por ejemplo, porque el funcionamiento del sistema no era tal y como lo esperaban.

Por tanto, una evaluación centrada en el usuario también es requerida. Ésta puede ser realizada "online" después de que el sistema haya sido lanzado, o como un estudio del usuario. Durante la evaluación "online", verdaderos usuarios actúan recíprocamente con el sistema sin ser conscientes del experimento que hay por detrás. Es posible controlar varias versiones de los algoritmos sobre diferentes grupos de usuarios para la comparación y el análisis del sistema con el objetivo de agrandar el sistema. Además, la mayor parte de los algoritmos incluyen parámetros, requiriendo el ajuste constante y la calibración.

Los estudios de usuario se llevan a cabo cuando la evaluación "online" no es factible o demasiado arriesgada. En este tipo de evaluación, se planifica un experimento controlado donde se pide a un pequeño grupo de usuarios realizar tareas diferentes con varias versiones del sistema. Entonces es posible analizar el comportamiento del usuario y distribuir cuestionarios de modo que los usuarios puedan hacer un informe sobre su experiencia. En tales experimentos, es posible reunir tanto información cuantitativa como cualitativa sobre los sistemas.

En los últimos años ha habido un interés creciente hacia los procedimientos de evaluación centrados en el usuario y métricas para los Sistemas de Recomendación. Los investigadores se dieron cuenta de que los objetivos de los SRs se extienden más allá de la exactitud de los algoritmos como instrumentos para proporcionar una experiencia útil y agradable que conduce a la retención del usuario y la satisfacción. Este criterio aumentó la gama de los aspectos evaluados de un RS para incluir aspectos como la forma de desencadenar las preferencias, la presentación de los resultados recomendados, y finalmente, la evaluación de las explicaciones proporcionadas por los usuarios. Las explicaciones pueden ayudar a varios objetivos: el más popular es la justificación de resultados, por ejemplo, explicando al usuario por qué el sistema decidió recomendar un artículo específico. Otros objetivos pueden incluir la confianza creciente en el sistema, convenciendo al usuario de comprar el ítem recomendado, y ayudando al usuario con su toma de decisiones. Diseñando la evaluación de la explicación de la recomendación, es importante identificar el objetivo de la explicación.

2.8 Librerías públicas para la definición de API REST

2.8.1 Dropwizard

Dropwizard[1] es un software libre que utiliza código Java para el desarrollo de REST backends con un alto rendimiento. Fue desarrollado por Yammer[2] para impulsar su backend basado en JVM (en inglés, Java Virtual Marchine). Dropwizard es un método disimulado de hacer rápidas aplicaciones web en Java. Reúne bibliotecas estables, librerías consistentes del entorno Java en un paquete simple y ligero que deja al usuario enfocarse en realizar bien el desarrollo del servicio web.

Dropwizard proporciona las mejores librerías Java en un paquete de aplicación ya integrado. Esto consiste en alguno de los siguientes componentes: Embedded Jetty, JAX-RS, JSON, Logging, Hibernate Validator, Metrics.

En este TFG, Dropwizard ha sido la elegida para la implementación del servicio web, en lugar de Spring, Play u otras herramientas para la construcción de API REST. Las razones principales de esta elección son las siguientes:

- Bootstrap. Rapidez a la hora de realizar el proyecto con Bootstrap, ya que bastará con añadir una dependencia al archivo del proyecto pom.xml y ya estaría listo.
- Métricas. Con Dropwizard tenemos el apoyo de las métricas. Esto proporciona una información muy útil como la petición o el tiempo de respuesta. Solamente tenemos que añadir la anotación @Timed para conseguir el tiempo de ejecución del método que queramos analizar en concreto.
- Productividad: Cada aplicación de Dropwizard tiene un programa principal que empieza el contenedor jetty. Esto significa que podemos ejecutar y debuggear la aplicación como un programa sin un entorno de desarrollo necesario. Además, no es necesario recompilar o reasignar el archivo web.
- Simplicidad: Si el usuario necesita algo, todo lo que éste tiene que hacer es añadir el jar o la dependencia y se puede usar. Tiene un ciclo de vida establecido para servicios controlados.

2.8.2 Play

El marco de Play hace fácil la construcción de aplicaciones web con Java y Scala. Play está basado en una arquitectura amigable, ligera y apátrida. Construido en Akka, Play proporciona un predecible y mínimo consumo de los recursos (CPU, memoria, hilos) para usos altamente escalables.

La versión Play 1.x. era simple y fácil de debuggear. Sin embargo, la versión Play 2.x introduce muchos rasgos mejores, de manera que es más completa. Sin embargo, Dropwizard recoge un puñado de lo mejor y lo junta en distintas clases de librerías.

[1] https://www.dropwizard.io/1.3.1/docs/

[2] Yammer es una herramienta para organizaciones privadas que se ha creado para poder comunicarse de forma abierta como una red social. https://www.yammer.com/

Play 2.x introduce cosas realmente buenas en aplicaciones streaming. Este rasgo es importante, ya que entonces es apto para su uso. Sin embargo, encontrar bibliotecas para trabajar en su marco requieren demasiado esfuerzo. **Apache Shiro** no trabaja con ellos por no tener un modelo de servlet. De manera que no puede ampliar las capacidades de un servidor. Hay alternativas, pero no se puede reutilizar código existente en otro proyecto.

2.8.3 Spring

Un elemento clave de Spring[3] es el soporte infraestructural a nivel de aplicación: Spring se centra en la parte más "laboriosa" de la aplicación de la empresa, de modo que los distintos departamentos puedan enfocarse en la lógica de negocio, sin depender innecesariamente de entornos de desarrollo específicos.

2.9 Librerías públicas para Recomendación

2.9.1 RankSys

La librería principal que se ha utilizado para generar recomendaciones es RankSys[4], un entorno que contiene funciones propias de algoritmos de recomendación y evaluación. RankSys ha sido el resultado de una investigación que actualmente está documentado y además forma parte de una tesis doctoral.

El código de esta librería ha sido programado en Java 8, que actualmente es la versión más reciente en este lenguaje de programación.

Para que el sistema pueda ofrecer recomendaciones a usuarios sobre ítems, se ha de entrenar el servidor con los datos existentes. Los métodos de recomendación implementados han sido de popularidad, similitud y factorización de matrices (todos ellos de filtrado colaborativo) [5][9].

2.9.2 LensKit

Otra librería interesante para la generación de recomendaciones es LensKit[5], pero no es la que se ha utilizado en este trabajo.

LensKit es un software libre que ofrece prácticamente las mismas características que RankSys, pero no se llega a utilizar ya que RankSys era el entorno recomendado por el tutor al tener más conocimiento sobre la librería. Además, LensKit es un poco más compleja de configurar y se ha utilizado en menos trabajos en la literatura.

[3] https://spring.io/
[4] http://ranksys.org/
[5] http://lenskit.org/

3 Diseño

3.1 Estructura de datos

Para el diseño de la API REST basada en un Sistema de Recomendación (el objetivo de este trabajo), se necesita un modelo de entidad relación bastante sencillo.

Nuestro modelo se basa en tres objetos principales: User, Item, Event [1]. El usuario tendrá un identificador, al igual que el ítem y el evento. De esta manera tendremos un atributo único para cada objeto. El evento tendrá además como atributos el identificador del usuario y el ítem al que hace referencia. Los objetos tendrán atributos adicionales que hacen que estén mejor definidos y sirvan para conseguir unas recomendaciones mejores, de acuerdo con el estudio de la literatura del capítulo anterior y que se podría extender en el futuro.

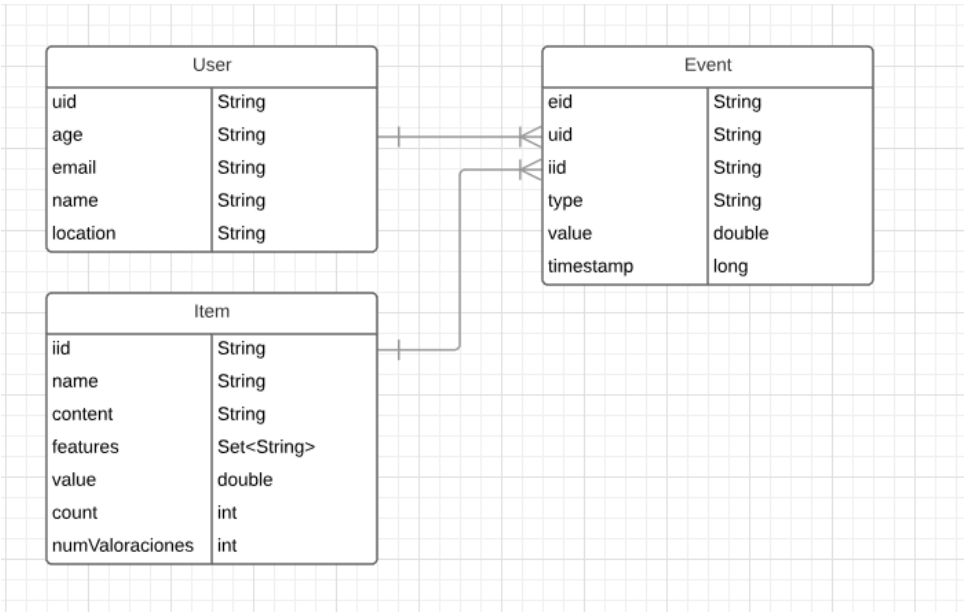

Ilustración 1. Modelo de estructura de datos[6]

3.2 Requisitos funcionales de la API REST para un Sistema de Recomendación

Para que la plataforma donde se utiliza un Sistema de Recomendación pueda recoger los datos que le interese conocer, se debe conocer la API REST con la que se está trabajando, de manera que debemos recoger la información necesaria para saber qué hace el servicio y cómo debemos de realizar la invocación al mismo.

Por tanto, la API REST debe proporcionar los siguientes requisitos para que el usuario pueda obtener el máximo rendimiento del Sistema de Recomendación:

[6] Este esquema se ha realizado con la herramienta LucidChart. https://www.lucidchart.com/

- Requisito funcional 1. La API debe permitir añadir un nuevo usuario.
- Requisito funcional 2. La API debe permitir ver los datos de un usuario.
- Requisito funcional 3. La API debe permitir ver todos los usuarios existentes en el sistema.
- Requisito funcional 4. La API debe permitir borrar un usuario.
- Requisito funcional 5. La API debe permitir obtener todos los eventos relacionados a un usuario.
    - Requisito funcional 6. La API debe permitir obtener los eventos de tipo "rating" relacionados a un usuario.
    - Requisito funcional 7. La API debe permitir obtener los eventos de tipo "count" relacionados a un usuario.
- Requisito funcional 8. La API debe ofrecer recomendaciones de ítems para un determinado usuario.
- Requisito funcional 9. La API debe permitir añadir un nuevo ítem.
- Requisito funcional 10. La API debe permitir ver la información de un ítem.
- Requisito funcional 11. La API debe permitir ver todos los ítems existentes en el sistema.
- Requisito funcional 12. La API debe permitir borrar un ítem.
- Requisito funcional 13. La API debe permitir obtener todos los eventos relacionados a un ítem.
    - Requisito funcional 14. La API debe permitir obtener los eventos de tipo "rating" relacionados a un ítem.
    - Requisito funcional 15. La API debe permitir obtener los eventos de tipo "count" relacionados a un ítem.
- Requisito funcional 16. La API debe permitir añadir un evento
    - Requisito funcional 17. La API debe permitir añadir un evento de tipo "rating" de un usuario sobre un ítem.
    - Requisito funcional 18. La API debe permitir añadir un evento de tipo "count" de un usuario sobre un ítem.
- Requisito funcional 19. La API debe permitir obtener la información de un evento.
    - Requisito funcional 20. La API debe permitir obtener la información de un evento a partir del id de usuario y el id del ítem.
- Requisito funcional 21. La API debe entrenar el Sistema de Recomendación según el contenido que haya en el sistema.
- Requisito funcional 22. La API debe permitir ver un resumen de los usuarios, ítems y eventos según el tipo que se encuentran actualmente en el sistema.
- Requisito funcional 23. La API debe ser capaz de leer la información que le manden a través del método POST, en formato JSON.
- Requisito funcional 24. La API debe ser capaz de proporcionar la información que le pidan a través del método GET, en formato JSON.

3.3 Requisitos no funcionales de la API REST para un Sistema de Recomendación

Además de definir los requisitos funcionales, se han de concretar cuáles son los no funcionales. Estos deben proporcionar las propiedades o cualidades que tiene la API REST.

Por ello, los requisitos no funcionales que debe tener esta API REST basada en Sistema de Recomendación son los siguientes:

- Requisito no funcional 1. Eficiencia. El sistema tiene que permitir que el usuario pueda realizar varias acciones por segundo, además de que la respuesta ofrecida por el servidor sea rápida y no se demore demasiado.
- Requisito no funcional 2. Usabilidad. El sistema debe ofrecer unas URLs sencillas e intuitivas para que el usuario pueda acceder al servidor rápidamente. El servidor tiene que tener una pequeña interfaz gráfica para poder realizar algunas operaciones de manera más simple que desde una terminal u otro acceso.
- Requisito no funcional 3. Mantenibilidad. El servidor tiene que ser capaz de estar activo correctamente y no se caiga al producirse un fallo en alguna petición de usuario.
- Requisito no funcional 4. Disponibilidad. El sistema tiene que tener una disponibilidad alta para que el usuario pueda acceder a él en todo momento. Además, la probabilidad de que el servidor falle debe ser muy baja.

3.4 Métodos API REST

Métodos
Para el manejo de esta API REST será suficiente con hacer llamadas al servidor con los métodos POST, GET y DELETE.

Con el método GET se obtiene información del servidor. Se recogen los datos que están en memoria en el servidor, o en una base de datos conectada a éste. No importa que para esto se tenga que enviar a través de la petición algún dato en concreto para obtener la respuesta del servidor esperada, como pueda ser en este caso obtener la información de un ítem en concreto, u obtener la información de todos los usuarios que se encuentran en el sistema. Si el sistema no puede recopilar la información solicitada, la respuesta llevará un código de error.

El método POST, al contrario que el GET, es para enviar información al servidor. El servidor será el encargado de procesar dicha información y agregarla o actualizarla en memoria o en la base de datos. En este servicio, cuando hacemos una llamada a POST, el servidor procesa los datos que le llegan a través de la petición, y devuelve información de relevancia en la respuesta, como, por ejemplo, al añadir un usuario, se devuelve el identificador asignado para dicho usuario.

El método DELETE, elimina información del servidor. El servidor recibe en la petición un identificador del objeto a eliminar, y elimina dicho elemento del servidor, así como toda su información, y en la respuesta devuelve un código de confirmación. Si el servidor no es

capaz de eliminar el elemento, porque, por ejemplo, el elemento a borrar no existe en el servicio, se devuelve un código de error en la respuesta.

4 Desarrollo

4.1 Servicio REST

Para la utilización del servicio REST, es muy importante definir una URL intuitiva y lógica para el usuario que lo vaya a utilizar. Por tanto, para cada función de la API se elige un nombre con sentido que vaya acorde a los que realiza internamente la función.

Las peticiones POST se realizan a través de la terminal ya que por URL no se puede mandar la información de un objeto por JSON. Las peticiones DELETE también se realizan desde la terminal, pero no por la misma razón que la petición POST, sino porque desde la URL no se puede detectar que es una petición de método DELETE, se entendería como una operación GET. Por último, las peticiones GET se pueden realizan también a través de la terminal como las anteriores, o podemos hacerlo desde la URL del navegador, donde este nos mostraría la respuesta que devolviera la petición.

La funcionalidad que ofrece el servicio REST están definidas en las siguientes subsecciones, indicando el método HTTP y su URL asociado, así como una breve descripción de cada funcionalidad. En todos los casos se indica como servidor en la URL localhost y el puerto el 8080, pero esto se puede configurar en el proyecto y dependerá de dónde se aloje la aplicación de Dropwizard.

4.1.1 Métodos de usuario

AÑADIR UN USUARIO
Método: POST **URL**: http://localhost:8080/user/add

Añade un nuevo usuario al servidor. Para poder realizar esta petición hay que mandar como parámetro un JSON con la información del usuario. Si la petición es procesada correctamente, la respuesta contendrá el ID del usuario que se ha añadido al sistema. Si la petición falla, bien porque el parámetro que se manda es incorrecto o no contiene valor o porque en la petición se incluye un ID ya existente, la respuesta contendrá un "-1" significativo de que se ha producido un error.

OBTENER UN USUARIO
Método: GET **URL:** http://localhost:8080/user/get/{uid}

Obtiene la información de un usuario del servidor. Para ello, deberemos pasar por la URL el ID del usuario del que queremos obtener la información. Si la petición es correcta, el servidor devolverá la información del usuario solicitado en formato JSON. Si, en cambio, el usuario que mandamos no existe, el servidor no devolverá nada.

OBTENER TODOS LOS USUARIOS
Método: GET **URL:** http://localhost:8080/user/get

Obtiene la información de todos los usuarios que se encuentran en el sistema. La petición obtendrá como respuesta la lista de usuarios del servidor en formato JSON, y en caso de no existir usuarios, la respuesta se muestra vacía.

ELIMINAR UN USUARIO
Método: DELETE **URL:** http://localhost:8080/user/delete/{uid}

Elimina un usuario del sistema y toda su información. Para poder realizar correctamente la acción debemos mandar por la URL el id del usuario que queremos eliminar. La petición devolverá como respuesta el código "OK" si se ha conseguido eliminar correctamente el usuario o "-1" si no se ha podido eliminar.

OBTENER EVENTOS DE UN USUARIO
Método: GET **URL:** http://localhost:8080/user/get/{uid}/events

Obtiene la información de los eventos relacionados a un usuario en concreto. Como parámetro en la URL debemos pasar el ID del usuario del que se quieren obtener sus eventos. La petición devolverá como respuesta la lista de dichos eventos si es que el usuario tiene algún evento relacionado, si no es así, se devolverá una lista vacía.

OBTENER RECOMENDACIONES PARA UN USUARIO
Método: GET **URL:** http://localhost:8080/user/get/{uid}/recommendations

Obtiene recomendaciones de ítems para un usuario concreto. Para realizar la petición se ha de pasar el ID del usuario sobre el que queremos obtener recomendaciones. La petición devuelve una lista de ítems que se ajustan a las características y gustos del usuario que hemos mandado a través de la URL.

4.1.2 Métodos de Ítem

AÑADIR UN ÍTEM
Método: POST **URL:** http://localhost:8080/item/add

Añade un nuevo ítem al servidor. Para poder realizar esta petición, al igual que la acción de añadir un usuario, hay que mandar como parámetro un JSON con la información del ítem. Si la petición se procesa bien, la respuesta contendrá el identificador del ítem que se ha añadido al sistema. Si la petición falla, bien porque el parámetro que se manda es incorrecto o no contiene valor o porque en el JSON de la petición se incluye un ID ya existente, la respuesta contendrá un "-1" significativo de que se ha producido un error.

OBTENER UN ÍTEM
Método: GET **URL:** http://localhost:8080/item/get/{iid}

Obtiene la información de un ítem del sistema. Para ello, deberemos pasar por la URL el identificador único del ítem del que queremos obtener la información. Si la petición es contiene un ID de ítem existente, el servidor devolverá la información de dicho ítem en formato JSON. Sin embargo, si el ítem que mandamos no existe, el servidor no devolverá nada.

OBTENER TODOS LOS ÍTEMS
Método: GET **URL:** http://localhost:8080/item/get

Obtiene la información de todos los ítems que se encuentran en el sistema. La petición obtendrá como respuesta procedente del servidor la lista de ítems con sus atributos en formato JSON, y en caso de no haber ningún ítem en el sistema, se muestra una lista de ítems vacía.

ELIMINAR UN ÍTEM
Método: DELETE **URL:** http://localhost:8080/item/delete/{iid}

Elimina un ítem del servidor y toda su información. Para efectuar esta petición, debemos pasar por la URL el id del ítem que queremos eliminar. La respuesta del servidor será el código "OK" si se ha conseguido eliminar correctamente el ítem, o el código "-1" si no se ha podido eliminar.

OBTENER EVENTOS DE UN ÍTEM
Método: GET **URL:** http://localhost:8080/item/get/{iid}/events

Obtiene la información de los eventos relacionados a un usuario en concreto. Como parámetro en la URL debemos pasar el ID del usuario del que se quieren obtener sus eventos. La petición devolverá como respuesta la lista de dichos eventos si es que el usuario tiene algún evento relacionado, si no es así, se devolverá una lista vacía.

4.1.3 Métodos de Evento

AÑADIR UN EVENTO
Método: POST **URL:** http://localhost:8080/event/add

Añade un nuevo evento al servidor. Para poder realizar esta petición se ha de mandar como parámetro un JSON con la información del evento, en la que se debe incluir el ID del usuario y el ID del ítem sobre el que se ejecuta el evento. Además, se debe incluir también el tipo de evento, según si ha sido de tipo "rating" (valoración) o de tipo "count" (visitas). Si la petición se procesa bien, la respuesta contendrá el identificador del evento que se ha añadido al sistema. Si la petición falla, bien porque el parámetro que se manda es incorrecto o no contiene valor, porque en el JSON de la petición se incluye un ID ya existente, o porque el usuario o ítem que se mandan en la petición no estén en el servidor, la respuesta contendrá un "-1" significativo de que se ha producido un error.

AÑADIR UN EVENTO
Método: POST **URL:** http://localhost:8080/event/add/user/{uid}/item/{iid}

Esta petición añade un evento al servidor como la anterior. La diferencia es que no hace falta meter los identificadores del usuario y del ítem en el JSON que se envía. Se pueden mandar dichos IDs a través de la URL. Se ha añadido esta manera ya que puede ser útil también para el usuario que utilice el servidor tener dos maneras viables de añadir un evento al sistema.

OBTENER EVENTOS ESPECÍFICOS
Método: GET **URL:** http://localhost:8080/event/get/user/{uid}/ítem/{iid}

Obtiene la información de un evento o más que tiene un usuario e ítem específicos. Para poder procesar esta petición, el usuario debe mandar por URL un ID de usuario y de ítem existentes. Si esto es así, el servidor devuelve en la respuesta una lista con los eventos relacionados a dicho usuario e ítem. Si no existe ningún evento asociado a ese usuario y ese ítem, o uno de los IDs que se mandan por la URL no existiera, se devolvería una lista vacía.

OBTENER UN EVENTO
Método: GET **URL:** http://localhost:8080/event/get/{eid}

Obtiene la información de un evento del sistema. Para ello, deberemos mandar por la URL el identificador del evento sobre el que queremos obtener la información. Si la petición contiene un ID de evento correcto y existente en el servidor, éste devolverá la información de dicho evento en formato JSON. Sin embargo, si el ID del evento que pasamos no existe, el servidor no devuelve nada.

OBTENER EVENTOS DE UN USUARIO
Método: GET **URL:** http://localhost:8080/event/get/user/{uid}

Obtiene la información de los eventos relacionados a un usuario en concreto. Al igual que en los métodos de usuario, se pasa como parámetro en la URL el ID del usuario del que se quieren obtener sus eventos. La petición devolverá como respuesta la lista de dichos eventos si es que el usuario tiene algún evento relacionado, si no es así, se devolverá una lista vacía. Se hace esta URL para que sea posible acceder a los eventos de usuario desde la URL de usuario y de evento.

OBTENER EVENTOS DE TIPO RATING DE UN USUARIO
Método: GET **URL:** http://localhost:8080/event/get/user/{uid}/rating

Obtiene la información de los eventos de tipo rating relacionados con un usuario en concreto. Para esto, se pasa como parámetro en la URL el ID del usuario del que se quieren obtener sus eventos de tipo rating. La petición devolverá como respuesta la lista de dichos eventos en los que el usuario ha realizado alguna valoración de un ítem, si es que el usuario tiene algún evento relacionado de este tipo, si no es así, se devolverá una lista vacía.

OBTENER EVENTOS DE TIPO COUNT DE UN USUARIO
Método: GET **URL:** http://localhost:8080/event/get/user/{uid}/count

Obtiene la información de los eventos de tipo count relacionados con un usuario en concreto. Para que el servidor procese esta información, se pasa como parámetro de la URL el ID del usuario del que se quieren obtener sus eventos de tipo count. La petición devolverá como respuesta la lista de dichos eventos en los que el usuario ha realizado alguna visita sobre un ítem cualquiera, si es que el usuario tiene algún evento relacionado de este tipo, si no es así, se devolverá una lista vacía.

OBTENER EVENTOS DE UN ÍTEM
Método: GET **URL:** http://localhost:8080/event/get/item/{uid}

Obtiene la información de los eventos relacionados a un ítem en concreto. Para realizar esta acción, se pasa como parámetro en la URL el ID del ítem del que se van a obtener sus

eventos relacionados. La petición devolverá como respuesta la lista de dichos eventos si es que el ítem tiene algún evento asociado, si no es así, se devolverá una lista vacía. Al igual que pasa con los eventos de usuario, también existen dos maneras de obtener los eventos de un ítem, para que sea viable y cómodo para el usuario que se conecte con el servidor acceder a esta información de dos maneras diferentes y correctas a su vez.

OBTENER EVENTOS DE TIPO RATING DE UN ÍTEM
Método: GET **URL:** http://localhost:8080/event/get/item/{iid}/rating

Obtiene la información de los eventos de tipo rating relacionados a un ítem en concreto. En esta acción, se pasa como parámetro de la URL el ID del ítem del que se quieren obtener los eventos en los que dicho ítem ha sido valorado. La petición devolverá como respuesta la lista de eventos de tipo rating en los que el ítem esté involucrado. Si no es así, se devolverá una lista vacía.

OBTENER EVENTOS DE TIPO COUNT DE UN ÍTEM
Método: GET **URL:** http://localhost:8080/event/get/item/{iid}/count

Obtiene la información de los eventos de tipo count relacionados con un ítem en concreto. En este método, de la URL se ha de incluir el ID del ítem del que se quieren obtener los eventos en los que el ítem ha sido por un usuario. La petición devolverá como respuesta la lista de eventos de tipo count en los que el ítem esté relacionado. Si no es así, se devolverá una lista vacía.

OBTENER TODOS LOS EVENTOS
Método: GET **URL:** http://localhost:8080/event/get

Obtiene la información de todos los eventos que se encuentran en el sistema. La petición obtendrá como respuesta procedente del servidor la lista de eventos que existen en el sistema con sus atributos en formato JSON. En el caso de no existir ningún evento, se muestra una lista de eventos en blanco.

ENTRENAR AL SISTEMA
Método: GET **URL:** http://localhost:8080/train

Entrena al sistema explotando toda la información recopilada hasta ese momento. Este método se utiliza internamente, para que el servidor “entrene”, esto quiere decir, que el servidor conozca los usuarios que tiene el sistema, los ítems que existen, así como los eventos entre unos y otros. De esta manera, si el servidor está entrenado, cuando se llame al método de generar recomendaciones, el servidor conocerá que ítems son más convenientes para cada usuario, según todo el historial de eventos que ha ocurrido anteriormente, y dichas recomendaciones serán más útiles y correctas. El servidor define que éste se entrena automáticamente cada vez que se añaden 100 eventos nuevos al sistema, no obstante, llamando a este método se fuerza que se vuelva a entrenar en un momento concreto.

OBTENER INFORMACIÓN DEL SERVIDOR
Método: GET **URL:** http://localhost:8080/statistics/get

Obtiene información resumida de los datos existentes en el servidor. La llamada a esta URL devuelve el número de usuarios existentes en el servidor, así como el número de

ítems y el número de eventos. También especifica cuántos eventos son de tipo rating y cuántos de tipo count.

4.2 Cliente

En un servicio REST, es muy importante para el cliente (el usuario final que utiliza el servicio) que sea un sistema fácil de utilizar e intuitivo. Para ello, contar con páginas web que permitan realizar funciones del servidor a partir de una interfaz es la mejor manera de conseguirlo.

En este TFG se ha implementado una interfaz para el usuario bastante simple que permitirá añadir usuarios e ítems al sistema, y ver los datos existentes en el servidor de usuarios, ítems y eventos, utilizando la API REST descrita en la sección anterior.

Para poder realizar estas páginas web, se han utilizado plantillas FreeMaker (archivos .ftl), que son las que utiliza la librería de DropWizard. Están plantillas utilizan un lenguaje HTML y no tienen ninguna complicación, siendo una extensión de las clásicas JSPs de los servlets de Java. De esta manera es muy sencillo implementar páginas que sirvan como interfaz mientras se tenga una buena estructura de paquetes dentro del proyecto.

Las URL del cliente que están implementadas en este servicio está definidas a continuación. Hay que destacar que, al igual que en las URLs del servicio REST, se asume que el cliente está instalado en localhost en el puerto 8080.

4.2.1 Métodos de interfaz

PÁGINA DE INICIO
Método: GET **URL:** http://localhost:8080/index

Esta URL muestra al cliente todas las acciones que se pueden realizar desde la interfaz. Las acciones se muestran como un listado, entre las que se encuentran: ver los usuarios del sistema; añadir un usuario al sistema; Ver ítems del sistema; Añadir ítems al sistema; Ver eventos del sistema. El cliente podrá elegir cualquiera de las opciones y ésta le redigirá a la página correspondiente. Esta página es simple administración de página para hacer más sencillo el uso del servicio al usuario.

OBTENER LISTADO DE USUARIOS
Método: GET **URL:** http://localhost:8080/interface/get/users

Obtiene el listado de usuarios existentes en el sistema. Esta página llama a la función del servidor que obtiene los usuarios que hay en un determinado momento. En la página se muestra el id del usuario, el nombre, y su localización, en caso de existir usuarios. Esta información se muestra a modo de tabla con toda la lista de usuarios. En caso de no existir usuarios en el sistema, solo se mostrarían las cabeceras de cada columna de la tabla, pero la tabla estará vacía.

FORMULARIO PARA AÑADIR USUARIO AL SISTEMA
Método: GET **URL:** http://localhost:8080/interface/add/user

Contiene un formulario para añadir un usuario al sistema. Esta página se muestra cuando el usuario quiere añadir un nuevo usuario el sistema. En la página se encontrará un formulario en el que habrá que rellenar la información del usuario. Se ha de rellenar el nombre, la localización, su email, y su edad. Una vez esté rellenada la información podremos pulsar el botón de añadir usuario. Una vez se haya completado la petición, y el servidor le haya asignado un id propio al usuario, el sistema redigirá al cliente a la página en la que se muestran todos los usuarios del sistema, entre los que se debe encontrar el nuevo usuario.

AÑADE UN USUARIO AL SISTEMA
Método: POST **URL:** http://localhost:8080/interface/add/user

Añade un nuevo usuario al sistema. En la página que contiene el formulario de añadir un usuario, se encuentra un botón para mandar la información al servidor. Dicho botón llama a esta función, que es la que procesa la petición, y la que crea el usuario con los datos que se hayan rellenado previamente. Esta función redirige a la página que muestra todos los usuarios del sistema.

OBTENER LISTADO DE ÍTEMS
Método: GET **URL:** http://localhost:8080/interface/get/items

Obtiene el listado de ítems existentes en el sistema. En esta página, al igual que en la de usuario se mostrará la información relacionada a cada ítem que se encuentre en el servidor. En formato tabla, se mostrará el id del ítem, su nombre, y su contenido (sector al que pertenece el ítem). Si no existiera ningún ítem en el sistema se mostraría la tabla vacía.

FORMULARIO PARA AÑADIR ÍTEM AL SISTEMA
Método: GET **URL:** http://localhost:8080/interface/add/item

Contiene un formulario para añadir un ítem al sistema. En esta página se debe rellenar la información adecuada para añadir un nuevo ítem al sistema. Para ello la interfaz muestra un formulario en el que se ha de llenar el nombre y el contenido del ítem. Una vez rellenado el formulario el usuario puede pulsar en el botón de añadir ítem, y una vez procesada la petición, se mostrará la lista de ítems existentes en el sistema.

AÑADE UN ÍTEM AL SISTEMA
Método: POST **URL:** http://localhost:8080/interface/add/item

Añade un nuevo ítem al sistema. Para procesar la petición de añadir un ítem desde la página que contiene el formulario, el botón que contiene llama a esta función. En la llamada pasaremos los datos rellenados por el usuario como parámetro, y la función se encarga de añadir el ítem al sistema asignándole un id único. Esta función se ha de encargar también del redireccionamiento a la página que muestra todos los ítems existentes el sistema, entre los que se debe encontrar este último.

OBTENER LISTADO DE EVENTOS
Método: GET **URL:** http://localhost:8080/interface/get/events

Obtiene el listado de eventos existentes en el sistema. Esta URL nos llevará a una página en la que encontraremos todos los eventos que han ocurrido en servidor. Esta información

se mostrará en una tabla que contendrá el id del evento, el id del usuario relacionado al evento, el ítem sobre el que ha ocurrido, y cuál ha sido el tipo de evento, si ha sido de tipo rating o count (visitas). Los eventos se mostrarán, en caso de existir alguno, como una lista dentro de la tabla.

5 Integración, pruebas y resultados

Para poder saber si el servidor está bien implementado, se han realizado dos tipos de pruebas. La primera, pruebas de caja negra, pruebas unitarias en las que se comprueba que el servidor responde correctamente a cada petición que se realiza. La segunda es la implementación de un cliente a través de una interfaz gráfica de manera que el usuario pueda ver de manera más intuitiva el correcto funcionamiento del servidor, y se le haga más ameno el uso del sistema.

5.1 Pruebas de caja negra

Para estas pruebas se ha utilizado la librería de junit, muy útil para realizar pruebas unitarias de caja negra. Gracias al método *assertEquals* que contiene dicha librería se pueden comprobar que los métodos POST y GET funcionan correctamente y devuelven la misma información sobre un mismo objeto.

Caso de prueba: AÑADIR UN USUARIO

Descripción: En esta prueba se testea añadir un usuario al sistema, y a continuación obtener sus datos y comprobar que el id obtenido es el mismo que el del usuario que se añadió.
Área funcional: Métodos de usuario.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de usuario.
Funcionalidad y resultado esperado: Una vez añadido el usuario, se obtendrá un id como respuesta, a continuación, se hará una petición GET sobre ella, y entonces, si el id de la respuesta de la primera petición POST y el id obtenido a través del GET es el mismo, el test estará correcto. Si estos IDs son distintos o algo no ha ido como debe, saltará una excepción.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: AÑADIR UN USUARIO DE FORMA INCORRECTA

Descripción: En esta prueba se testea añadir un usuario de manera incorrecta en el sistema, y que el usuario no se añada por tanto al servidor.
Área funcional: Métodos de usuario.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de usuario, y se inicializa un usuario a null.
Funcionalidad y resultado esperado: Teniendo de primeras un usuario vacío, se intenta añadir al sistema, y la respuesta debería devolver el código -1 propio de un error. Después, se añadirá un usuario de manera correcta al servidor, y se vuelve a intentar añadir un usuario con ese mismo id del usuario añadido anteriormente, y también debe devolverse el código -1.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: OBTENER UN USUARIO

Descripción: En esta prueba se testea obtener un usuario después de haberlo añadido al sistema.
Área funcional: Métodos de usuario.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de usuario.
Funcionalidad y resultado esperado: Primero se añade un usuario de manera correcta, y una vez recibida la respuesta, obtenemos a partir del método GET su información. Si el usuario que se ha añadido, y el que se ha obtenido es idéntico, la prueba estará superada.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: ELIMINAR UN USUARIO

Descripción: En esta prueba se testea eliminar un usuario del sistema, y que no queda su información guardada en el servidor.
Área funcional: Métodos de usuario.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de usuario.
Funcionalidad y resultado esperado: Lo primero que se hace es añadir un usuario de forma correcta al sistema, y a continuación eliminarlo. Si se intenta obtener el usuario eliminado, se debería obtener una respuesta vacía, y en ese caso, el test estará superado.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: AÑADIR UN ÍTEM

Descripción: En esta prueba se testea añadir un ítem al sistema, y a comprobar que ha sido añadido correctamente y sus datos están en servidor.
Área funcional: Métodos de ítem.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de ítem.
Funcionalidad y resultado esperado: Se añade un ítem al sistema a través del método POST. Una vez se tiene la respuesta se le pide al servidor la información del ítem a partir del id obtenido en la respuesta del POST. Si la respuesta de la primera petición y el id del ítem obtenido después es el mismo, la prueba es correcta.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: AÑADIR UN ÍTEM DE FORMA INCORRECTA

Descripción: En esta prueba se testea añadir un ítem de manera incorrecta en el sistema, y que el ítem no se añada por tanto al servidor.
Área funcional: Métodos de ítem.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de ítem, y se inicializa un ítem a NULL.
Funcionalidad y resultado esperado: Teniendo un ítem vacío, se intenta añadir dicho ítem al sistema, y este debería devolver el código de fallo. Si pasa este primer filtro, se añade ahora de manera correcta un ítem, y al momento se vuelve a añadir un ítem con el id

obtenido en la petición anterior. Esta acción debe devolver un código de error ya que existe ese mismo ítem en el servidor.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: OBTENER UN ÍTEM

Descripción: En esta prueba se testea obtener un ítem después de haberlo añadido al sistema.
Área funcional: Métodos de ítem.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de ítem.
Funcionalidad y resultado esperado: Se añade un ítem al servidor, y a partir de la respuesta obtenida se intenta obtener su información. Si la información es la misma y comparten el mismo identificador, la prueba está superada.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: ELIMINAR UN ÍTEM

Descripción: En esta prueba se testea eliminar un ítem del sistema, y que no queda su información guardada en el servidor.
Área funcional: Métodos de ítem.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de ítem.
Funcionalidad y resultado esperado: A través del método POST, se añade un nuevo ítem que no existía en el servidor, si a continuación lo eliminamos del servidor, e intentamos obtener su información a partir del método GET, éste debe devolver un elemento vacío. Si es así, la prueba es correctamente superada.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: AÑADIR UN EVENTO DE TIPO RATING

Descripción: En esta prueba se testea añadir un evento de tipo rating al sistema, y a comprobar que ha sido añadido correctamente y sus datos están en servidor.
Área funcional: Métodos de evento.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de evento.
Funcionalidad y resultado esperado: Lo primero que realiza este test, es añadir un usuario y un ítem al servidor. Una vez creados y dentro del sistema, se añade dicha información al evento que vamos a añadir al sistema, además de definir que es de tipo rating y el valor de dicho rating. Se añade el evento, y a continuación comprobamos que, obteniendo la información del evento a partir de la respuesta, se trata del mismo evento. Si se trata del mismo evento, la prueba está superada.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: AÑADIR UN EVENTO DE TIPO COUNT

Descripción: En esta prueba se testea añadir un evento de tipo count al sistema, y a comprobar que ha sido añadido correctamente y sus datos están en servidor.
Área funcional: Métodos de evento.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de evento.
Funcionalidad y resultado esperado: Lo primero antes de todo es añadir un usuario y un ítem al servidor, para que sea posible el añadir el evento. Una vez creados y dentro del sistema, se añade dicha información al evento que vamos a añadir, y se define que es de tipo count. Se añade el evento, y a continuación comprobamos que, obteniendo la información del evento a partir de la respuesta, se trata del mismo evento. Si se trata del mismo evento, la prueba está superada.
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

Caso de prueba: AÑADIR UN EVENTO DE FORMA INCORRECTA

Descripción: En esta prueba se testea añadir un evento de manera incorrecta en el sistema, y que el evento no se añada por tanto al servidor.
Área funcional: Métodos de evento.
Datos y acciones de entradas: Se manda como parámetro el objeto JSON de la clase sobre la que se hace la prueba, en este caso el de evento, y se inicializa un evento a NULL.
Funcionalidad y resultado esperado: Al llegar un evento vacío, si se intenta añadir al sistema, éste debe devolver un código de error. Lo próximo será añadir un usuario al sistema y relacionarlo a un nuevo evento e intentar añadirlo. Este método tampoco es posible ya que al evento le sigue faltando un ítem relacionado, por tanto, el servidor debe devolver el código de error. A continuación, se añade un ítem al sistema y se relaciona al evento, de manera que ahora sí se puede añadir el evento de forma correcta al servidor. Por último, se intenta añadir el mismo evento que se acaba de incluir en el servidor, y esto debe dar fallo, ya que no deben existir dos eventos con el mismo identificador,
Requerimiento de ambiente de pruebas: El requerimiento principal es que el servidor ha de estar activo para poder procesar todas las peticiones.

5.2 Pruebas de interfaz

Además de los test unitarios, se ha creado una pequeña interfaz de prueba que sirva como concepto de la funcionalidad que ofrece el sistema al usuario. En la interfaz no se encuentran todas las acciones que la API REST contiene, pero sirve para testear algunas de ellas y ver una primera versión de una interfaz en la que el usuario se pueda sentir más cómodo y tenga una mayor usabilidad.

5.2.1 Índice

En esta pestaña encontramos todas las acciones que puede realizar el usuario desde la interfaz. Dichas acciones se muestran en un listado entre las que se encuentran ver los usuarios del sistema, añadir usuarios al sistema, ver ítems del sistema, añadir ítem al

sistema, y ver eventos al sistema. Esta página hace la función de ser la página principal de la interfaz y desde la que se puede acceder a cualquier página existente en la interfaz.

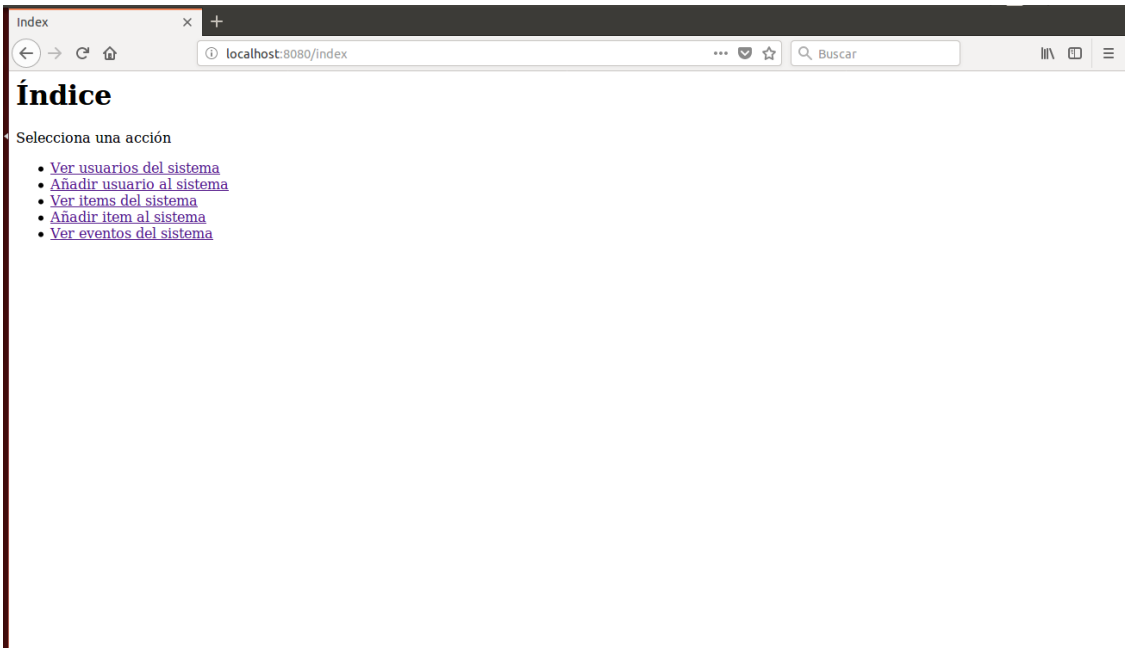

Ilustración 2. Índice de la interfaz

5.2.2 Lista de usuarios

En esta pestaña se ve el listado de todos los usuarios existentes en el sistema. En la imagen se muestran un ejemplo de varios usuarios “test” con distintos identificadores. Se muestra el ID del usuario, su nombre, y su localización. Además, se encuentra un enlace de acceso al índice para volver a la página principal.

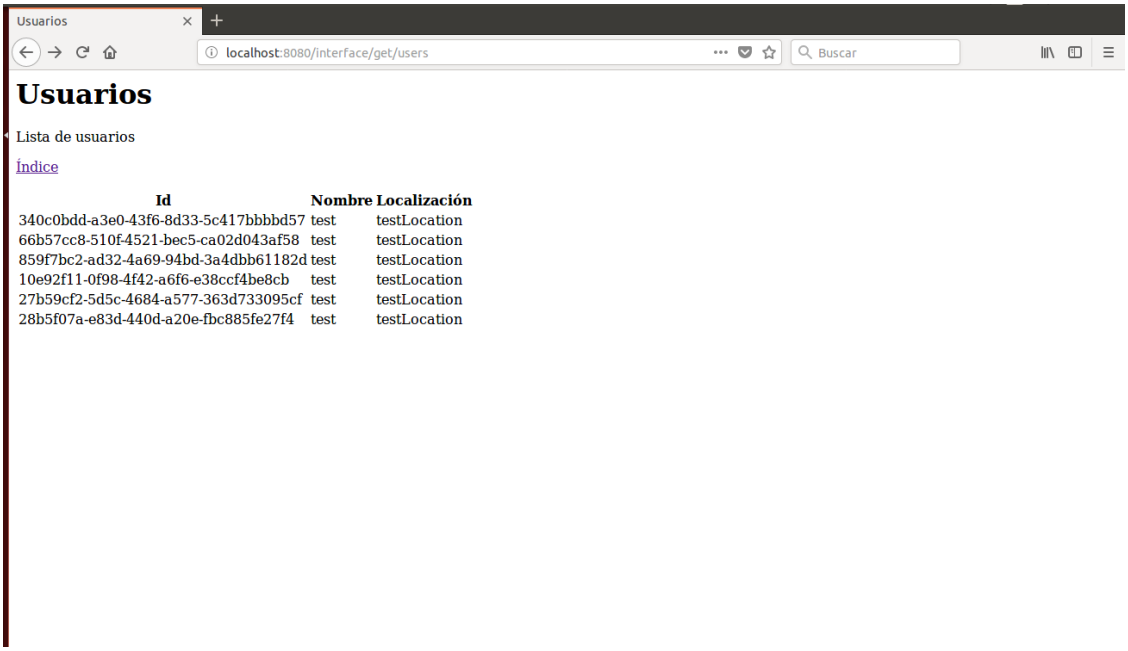

Ilustración 3. Lista de usuarios

5.2.3 Añadir usuario

En esta página, se muestra un formulario en el que se deben rellenar los datos del usuario que se quiere agregar al servidor. En este formulario se ha de rellenar su nombre, la localización, el email y la edad. Una vez estén rellenos estos campos, se tiene que pulsar el botón de añadir usuario.

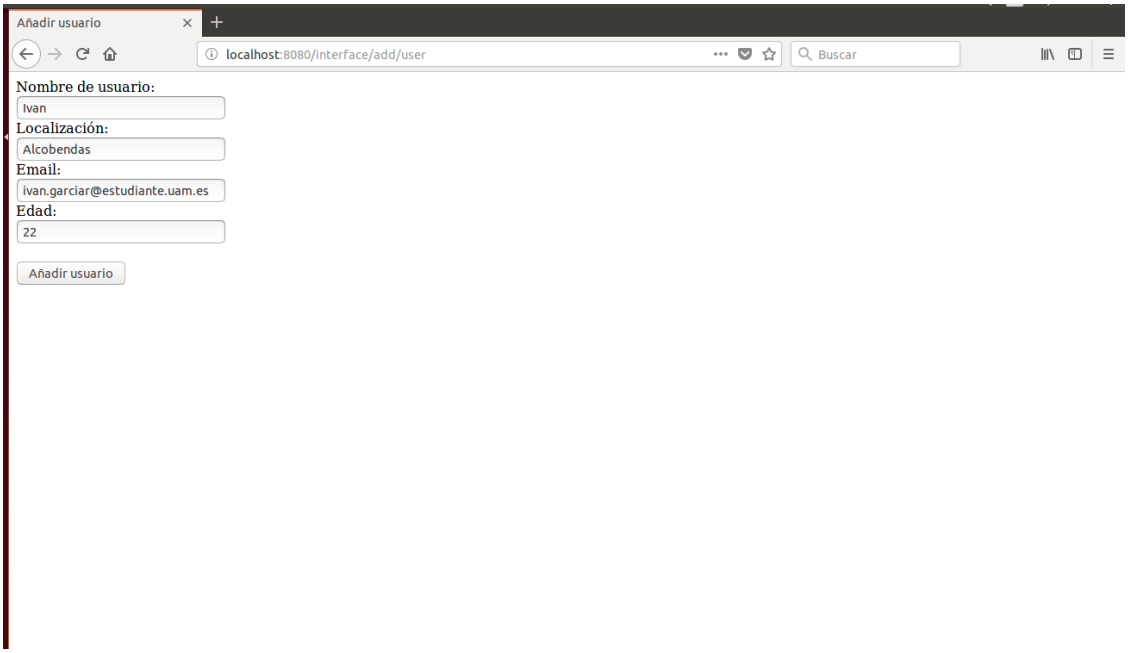

Ilustración 4. Interfaz para añadir un usuario

Una vez pulsado el botón, éste llevará a la página que muestra el listado de los usuarios del sistema, en el que se encuentra el nuevo usuario.

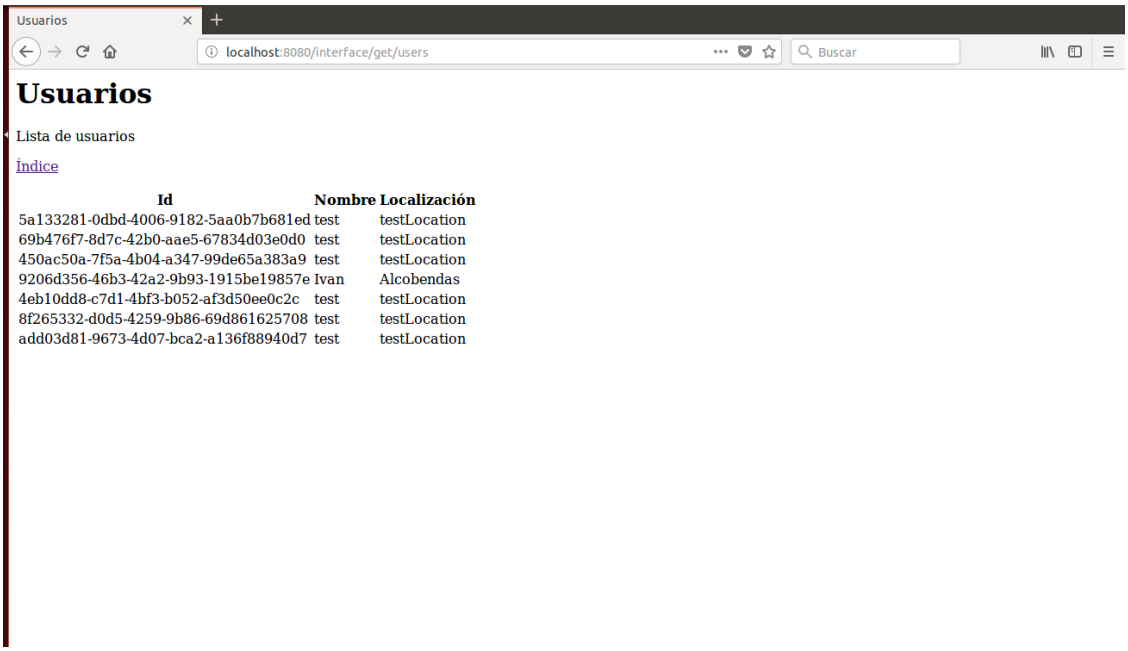

Ilustración 5. Lista de usuarios con el nuevo usuario

5.2.4 Lista de ítems

En esta pestaña se ve el listado de todos los ítems existentes en el sistema. En la página se muestran el identificador del ítem, su nombre y su contenido. Se ve que, aunque poseen el mismo nombre y mismo contenido, el identificador de cada ítem es distinto. También se puede encontrar un enlace a la página principal del sistema.

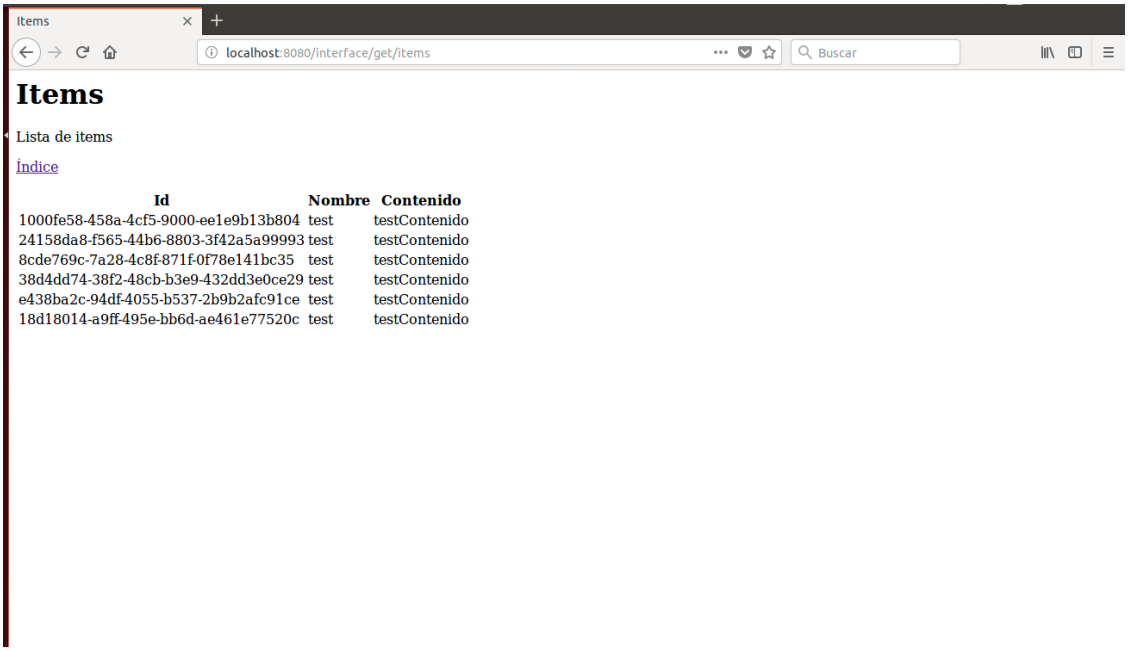

Ilustración 6. Lista de ítems

5.2.5 Añadir ítem

En esta página, se muestra un formulario en el que se deben rellenar los datos del ítem que se quiere añadir al servidor. En este formulario se ha de rellenar su nombre y su contenido. Una vez estén rellenos estos campos, se tiene que pulsar el botón de añadir ítem.

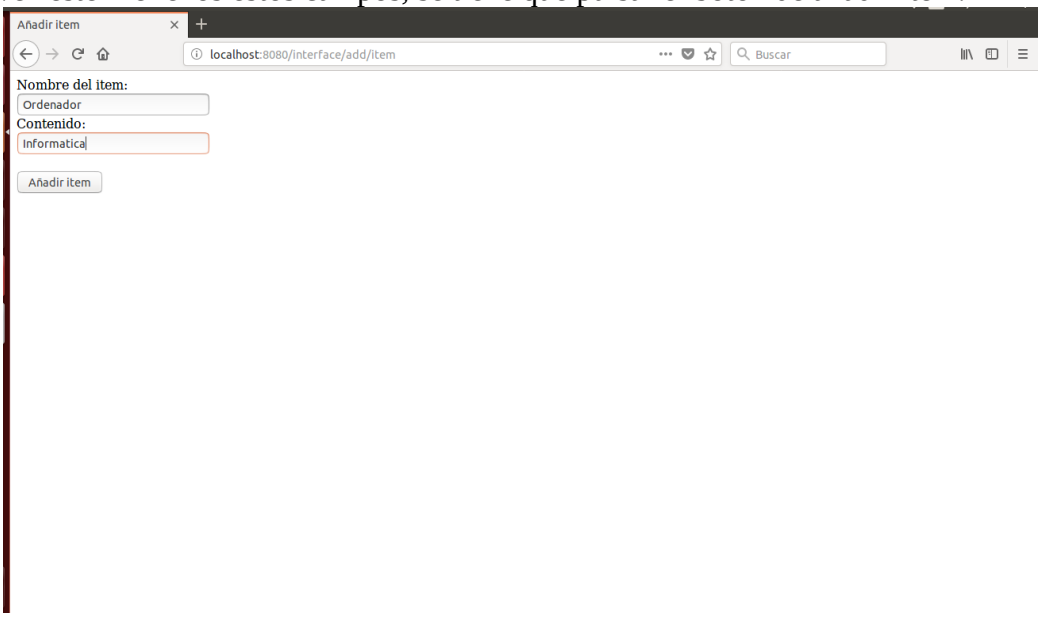

Ilustración 7. Interfaz para añadir un ítem

Una vez pulsado el botón, se redigirá al usuario a la página en la que se muestran todos los ítems del sistema, y se podrá ver que se encuentra el nuevo ítem.

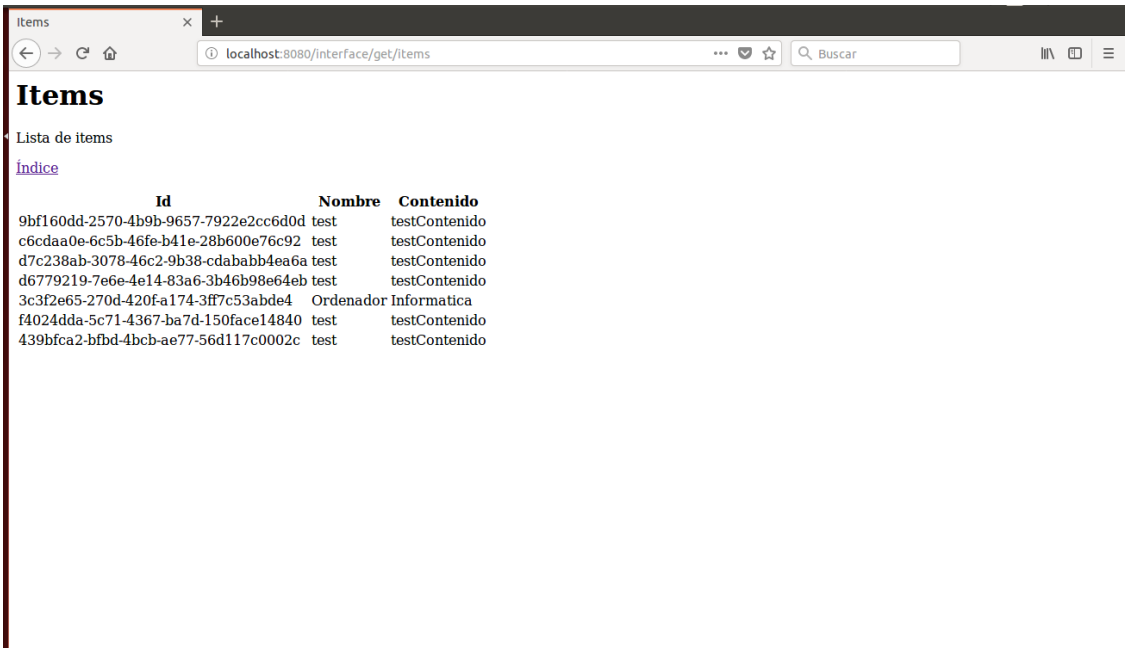

Ilustración 8. Lista de ítems con el nuevo ítem

5.2.6 Lista de eventos

En esta pestaña se ve el listado de todos los eventos existentes en el sistema. En la página se muestra una tabla con el ID del evento, el identificador del usuario, el identificador del ítem, y el tipo de evento. En la página se encuentra un enlace a la página principal del sistema.

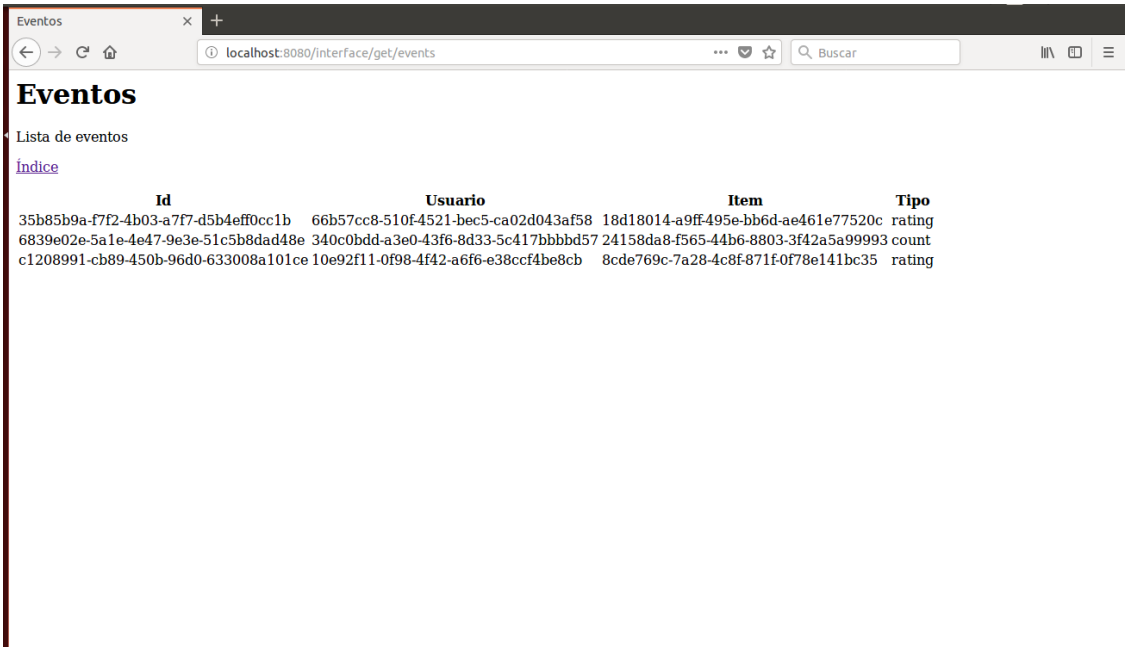

Ilustración 9. Lista de eventos

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este TFG se ha descrito en qué consisten los sistemas de Recomendación, y el impacto que tienen actualmente estos interesantes servicios para el mercado, como en comercio electrónico, la reproducción de videos online o el servicio online de música. Todo esto con el fin de generar un mayor número de ventas o la captación de un mayor número de usuarios en un sistema al ofrecer justo lo que el usuario desea.

Además, se ha implementado una pequeña versión de una API REST que puede ser utilizada por un Sistema de Recomendación. Esta API es un servicio web básico, pero que cubre las necesidades necesarias para poder ofrecer recomendaciones al usuario. La técnica de recomendación que se ha desarrollado es la del filtrado colaborativo.

Gracias a esto, se ha aprendido la utilización de librerías interesantes para el desarrollo de un servicio web, y librerías para ofrecer recomendaciones en un sistema.

Para el servicio web, la experiencia con DropWizard ha sido muy satisfactoria, ya que ayuda a probar y conectar el servidor de manera muy eficiente, y para las recomendaciones, RankSys ha sido la herramienta utilizada ya que era la recomendada por el tutor, y es una herramienta capaz de ofrecer recomendaciones bastante fiables.

La experiencia en este TFG ha sido completa ya que se ha aprendido a implementar un servicio web, con un soporte de una interfaz gráfica, en la que se han utilizados plantillas FreeMaker, a través de archivos .ftl.

Por último, decir que el aprendizaje sobre el mundo de los Sistemas de Recomendación ha resultado ser muy interesante ya que es algo que se utiliza en el día a día y se puede obtener un gran potencial con esta tecnología, que tiene como fin ayudar al usuario en su experiencia del uso de internet, intentando hacer un uso más directo e intuitivo.

6.2 Trabajo futuro

El trabajo presentado en este TFG es una primera versión de una API REST basada en un Sistema de Recomendación, pero se puede seguir trabajando en él y alcanzar muchas más mejoras para el usuario y encontrarle una mayor utilidad al sistema implementado.

- Interfaz gráfica. La interfaz gráfica implementada es una primera prueba de concepto. Por ello, esta interfaz podría tener una gran mejora, de manera que se puedan realizar todas las acciones permitidas por el servidor desde la interfaz, además de tener un diseño mejorado añadiéndole estilo y forma a la página.
- Integración. La API podría ser integrada en una aplicación web que ofrezca un servicio a usuarios.
- Interfaz “responsive”. Se podría realizar un diseño que fuera aplicable a distintos tamaños de dispositivos.

- Persistencia. La API ahora mismo guarda los datos en memoria. Un gran avance para la API REST sería guardar los datos en una base de datos, y así tener permitir que se mantenga la información incluso cuando el servidor se tiene que reiniciar.

Referencias

[1] Common Recommender REST API, http://www.recsyswiki.com/wiki/Common_Recommender_REST_API, recuperada gracias a https://web.archive.org/ con la URL https://web.archive.org/web/20161031103620/http://www.recsyswiki.com/wiki/Common_Recommender_REST_API.

[2] J. Bobadilla, F. Ortega, A. Hernando, A. Gutierrez, "Recommender systems survey. Knowledge-Based Systems", 2013.

[3] D. Bridge, M. Göker, L. McGinty, B. Smyth, "Case-based recommender Systems. The Knowledge Engineering Review", 2006.

[4] J. Castro Gallardo, "Un nuevo modelo ponderado para Sistemas de Recomendación Basados en Contenido con medidas de contingencia y entropía", Trabajo Tutelado de Iniciación a la Investigación, Universidad de Jaén, septiembre 2012.

[5] A. Gil Hernán, "Framework orientado a algoritmos de recomendación basados en vecinos cercanos", Trabajo Fin de Grado, Escuela Politécnica Superior, Universidad Autónoma de Madrid, febrero 2017.

[6] D. Goldberg, D. Nichols, B.M. Oki, D. Terry, "Using collaborative filtering to weave an information tapestry. Commun. ACM", 1992.

[7] A. Gunawardana, G. Shani, "Recommender Systems Handbook, 1st edition, Springer, Chapter 8. Evaluating Recommender Systems", 2011.

[8] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, "Evaluating Collaborative Filtering Recommender Systems", 2004.

[9] Y. Koren, R. Bell, "Recommender Systems Handbook, 1st edition, Springer, Chapter 3. Advances in Collaborative Filtering", 2011.

[10] M. Montaner, B. López, J.L. de la Rosa, "A taxonomy of recommender agents on the internet. Artificial Intelligence Review", 2003.

[11] F. Ricci, L. Rokach, B. Shapira, "Recommender Systems Handbook, 1st edition, Springer, Chapter 1. Recommender Systems: Introduction and Challenges", 2011.

Glosario

Def. Cliente web. Un cliente web, es cualquier aplicación o interfaz que tiene como finalidad utilizar recursos de la web. Un buen ejemplo serían los navegadores de Internet.

Def. E-commerce. El e-commerce se basa principalmente en la compra y venta de productos o servicios a través de internet.

Def. Ítem. Ítems (artículos) de un sistema de recomendación son aquellos objetos que son recomendados. Los artículos pueden ser caracterizados por su complejidad y por su valor o utilidad. El valor de un artículo puede ser positivo si el artículo es útil para el usuario, o negativo si éste no es apropiado y el usuario hizo la decisión incorrecta seleccionándolo. Notamos que cuando un usuario adquiere un ítem, siempre incurrirá en un coste que incluye el coste cognoscitivo de buscar el articulo y el verdadero coste monetario que se pagó por el artículo.

Def. REST. REST consiste en un tipo de arquitectura de desarrollo web que utiliza el lenguaje HTML. REST permite crear y desarrollar servicios web y aplicaciones que pueden ser entendidas por cualquier usuario que utilice o entienda HTML. REST es la arquitectura idónea para crear APIs que quieren ofrecer servicios orientados a Internet.

Def. Servidor web. Un servidor web es un software que procesa las peticiones de un usuario a través de una aplicación por el lado del servidor. El fin es realizar conexiones con el cliente ya sean síncronas o asíncronas y generando una respuesta en el lado del cliente. Para esta comunicación suele utilizarse algún protocolo de comunicación.

Def. Transacción. Generalmente, una transaction (transacción) o evento es una interacción registrada entre un usuario y el Sistema de Recomendación. Las transacciones son una estructura de datos que almacenan la información más importante generada durante la interacción entre el humano y la máquina, y que son útiles para el algoritmo de generación de recomendaciones que usa el sistema. Por ejemplo, una estructura de transacción puede contener una referencia al artículo seleccionado por el usuario y una descripción del contexto para esa recomendación en particular. Si está disponible, dicha transacción también puede incluir la regeneración explícita que el usuario ha proporcionado, como la valoración para el ítem seleccionado.

Def. URL. Uniform Resource Locator. URL es la dirección especifica de un recurso que se encuentra en Internet. Cualquier usuario puede localizar dichos recursos a través de las URLs.

Def. Usuario. Los users (usuarios) de un SR pueden tener objetivos muy diversos y características diferentes. Para personalizar las recomendaciones y la interacción entre el humano y la máquina, los Sistemas de Recomendación explotan una gama de información sobre los usuarios. Esta información puede ser estructurada de varios modos, y como se ha explicado en el Capítulo 2, la selección de qué información modelar depende de la técnica de recomendación.